Lab 4: HW/SW Compression and Decompression of the Captured Image

Objectives:

- Understand how to leverage the FPGA based hardware acceleration in an SoC design.

- Understand how communication is achieved between the FPGA fabric and the ARM hard core processor system (HPS) in an SoC board

- Understand how to add new components to an existing QSYS design to implement new functionalities

Tools:

1. **Quartus Prime** – Hardware design and compilation
    a. **QSYS** – For adding new components (PIOs) to the given design which facilitate communication between ARM HPS and Verilog RLE

2. **Altera Monitor Program** – Used for compiling, loading and debugging your program (image capture, B&W conversion, transfer image to RLE, decompression, compression ratio display) for ARM HPS on the DE1-SoC board

Detailed Procedure:

In Quartus Prime

1. For this lab, you shall modify the HW/SW files provided in lab 2 to implement new functionalities. You can either re-download the files from the website or re-use the files in your lab 2 directory.

2. Launch Quartus Prime, open the DE1_SoC_With_D5M QPF file present in the verilog folder. Download the FIFO buffer (rle_fifo_8_24.v) from the course website. Add the RLE that you designed and the FIFO buffer to your project. Open the Computer_System.qsys in QSYS.

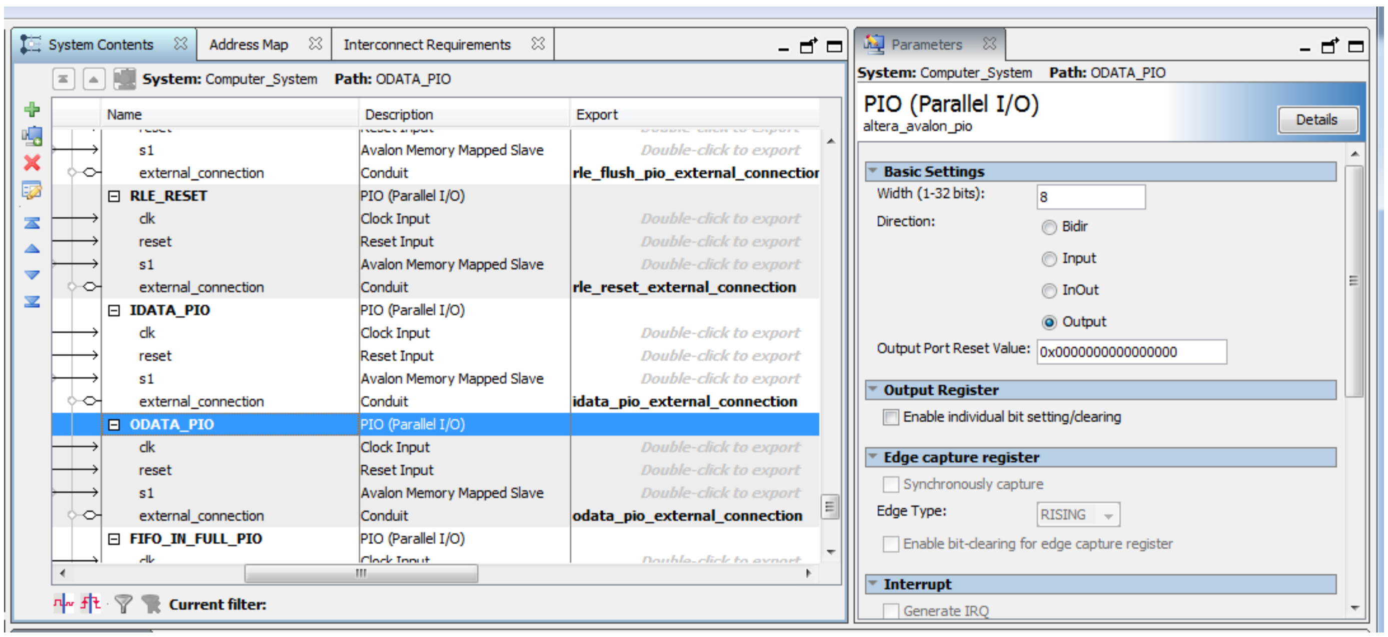
3. Add eight new Parallel Inputs/Outputs (PIOs) for the connections described in the slides (follow the same order). The PIO component can be found in the Library window of the QSYS. Rename each PIO to the names given in the slide below. The information in the round brackets is to tell you whether the signal is input or output. **Don't assign the base address after adding each PIO.** Base address assignment must be done after adding and making the necessary connections.

Signals in Block Structure (To be added in QSYS)

- RESULT_READY_PIO (Input)
  - Indicates that there is an encoded data segment in the FIFO. Note that this signal is active low since it is tied to the FIFO empty output.
- RLE_FLUSH_PIO (Output)
  - Used at the end of the bit-stream. RLE produces the final encoded data segment immediately with the last counting bit value.
- RLE_RESET (Output)
  - Signal for initializing RLE encoder. Assert and de-assert at the beginning of the program.
- IDATA_PIO[23:0] (Input)
  - Input ports to receive the encoded data.
  - 1 bit for bit ID, 23 bits for representing number of bits.
- ODATA_PIO[7:0] (Output)
  - Output ports to send original bit-stream. Data is sent in 8-bit segments.
- FIFO_IN_FULL_PIO (Input)
  - Indicates FIFO is full. Sending picture data stream should wait until this signal is de-asserted.
- FIFO_IN_WRITE_REQ_PIO (Output)
  - Asserted to write bit-stream segment to FIFO in buffer. FIFO stores input data when this signal is asserted.
- FIFO_OUT_READ_REQ_PIO (Output)
  - Asserted when ARM wishes to read from the FIFO out. FIFO produces next data from the buffer when this signal is asserted.

4. Make sure that each PIO has the correct parameters. Set the direction to input or output in relation to the ARM core, and set the bit-width to match the bit-width of the signal. For example, ODATA_PIO is an 8-bit output.

5. In the external_connection – Conduit row, find the export column and double click to export the signal. The default name used for exporting should be the PIO name followed by _external_connection. For example, ODATA_PIO corresponds to odata_pio_external_connection.


System Contents | Address Map | Interconnect Requirements

System: Computer_System **Path:** ODATA_PIO

Name	Description	Export
s1	Avalon Memory Mapped Slave	Double-click to export
external_connection	Conduit	rle_flush_pio_external_connectior
RLE_RESET	PIO (Parallel I/O)	
clk	Clock Input	Double-click to export
reset	Reset Input	Double-click to export
s1	Avalon Memory Mapped Slave	Double-click to export
external_connection	Conduit	rle_reset_external_connection
IDATA_PIO	PIO (Parallel I/O)	
clk	Clock Input	Double-click to export
reset	Reset Input	Double-click to export
s1	Avalon Memory Mapped Slave	Double-click to export
external_connection	Conduit	idata_pio_external_connection
ODATA_PIO	PIO (Parallel I/O)	
clk	Clock Input	Double-click to export
reset	Reset Input	Double-click to export
s1	Avalon Memory Mapped Slave	Double-click to export
external_connection	Conduit	odata_pio_external_connection
FIFO_IN_FULL_PIO	PIO (Parallel I/O)	

Current filter:

Parameters

System: Computer_System **Path:** ODATA_PIO

PIO (Parallel I/O)
altera_avalon_pio

Details

Basic Settings

Width (1-32 bits): 8

Direction: Bidir [✓ Output] Input InOut

Output Port Reset Value: 0x0000000000000000

Output Register

Enable individual bit setting/clearing

Edge capture register

Synchronously capture

Edge Type: RISING

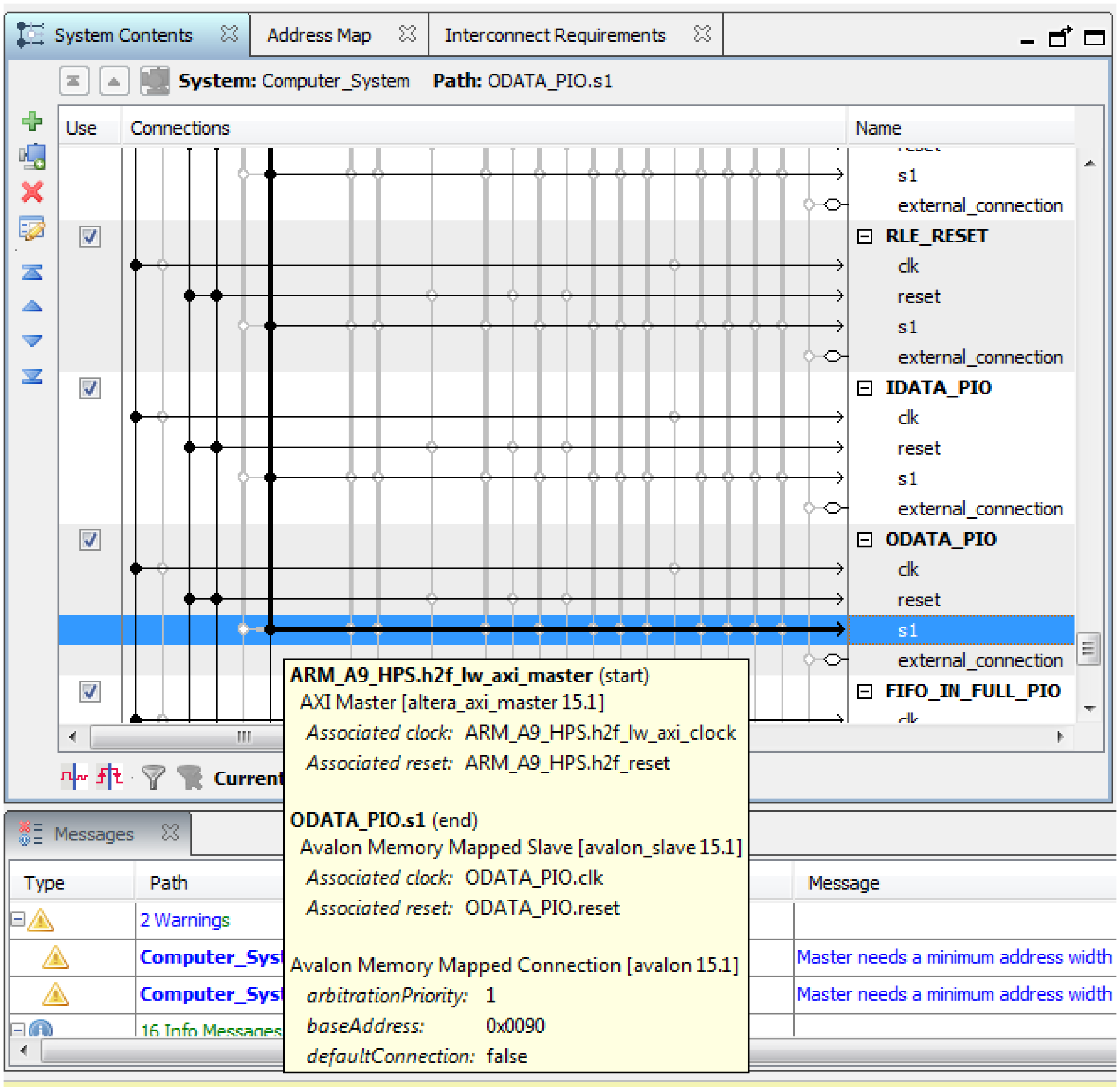
Enable bit-clearing for edge capture register

Interrupt

Generate IRQ

6. Connect each PIO to the clock signals, reset signals, and HPS-FPGA lightweight bridge as in the following screenshot.

a. clk – System_PLL.sys_clk
b. reset – System_PLL.reset_clk and ARM_A9_HPS.h2f_reset
c. s1 – ARM_A9_HPA.h2f_lw_axi_master



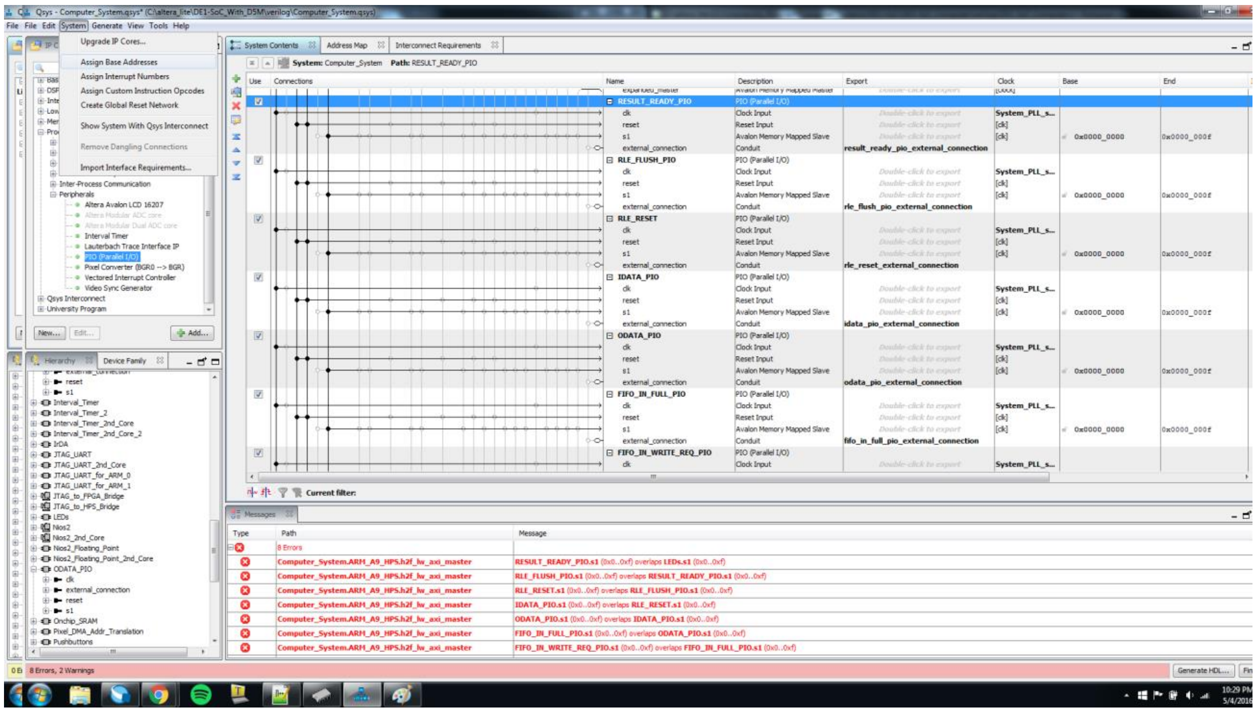
ARM_A9_HPS.h2f_lw_axi_master (start)
AXI Master [altera_axi_master 15.1]
Associated clock: ARM_A9_HPS.h2f_lw_axi_clock
Associated reset: ARM_A9_HPS.h2f_reset

ODATA_PIO.s1 (end)
Avalon Memory Mapped Slave [avalon_slave 15.1]
Associated clock: ODATA_PIO.clk
Associated reset: ODATA_PIO.reset

Avalon Memory Mapped Connection [avalon 15.1]
arbitrationPriority: 1
baseAddress: 0x0090
defaultConnection: false

Type	Path	Message
⚠	2 Warnings	
⚠	Computer_Syst	Master needs a minimum address width
⚠	Computer_Syst	Master needs a minimum address width
ⓘ	16 Info Messages	

7. You will see a number of errors in the messages segment relating to address overlaps. Fix these errors by selecting Assign Base Addresses in the System menu.



Qsys - Computer_System.qsys* (C:\altera_lite\DE1-SoC_With_D5M\verilog\Computer_System.qsys)

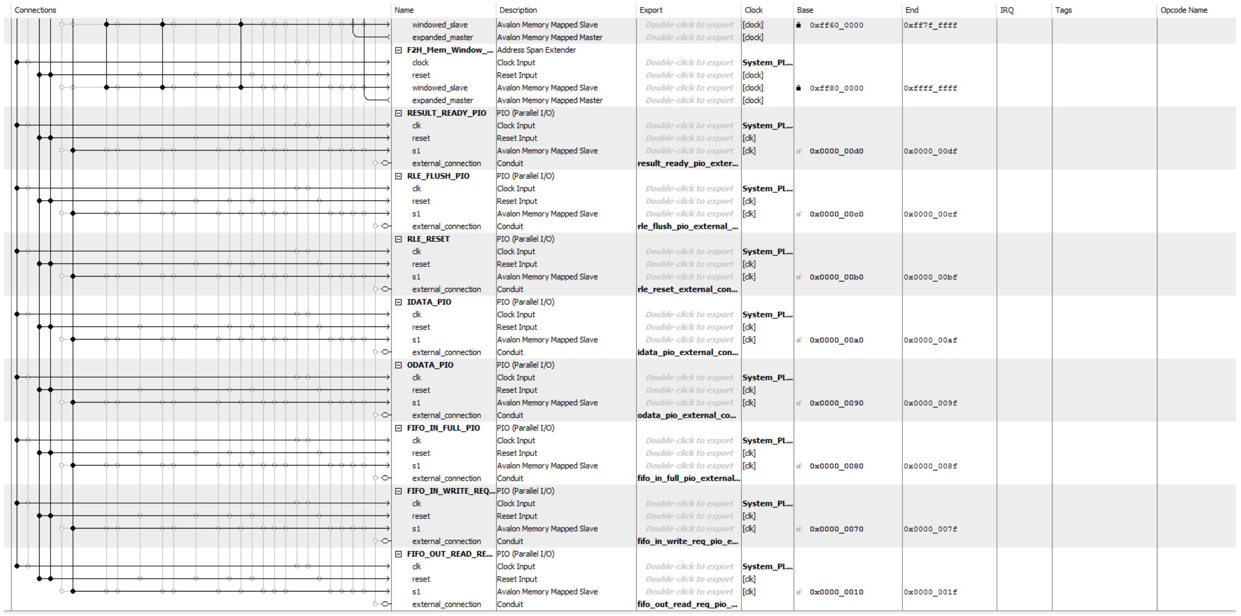
System: Computer_System Path: RESULT_READY_PIO

Name	Description	Export	Clock	Base	End
⊟ RESULT_READY_PIO	PIO (Parallel I/O)				
clk	Clock Input	Double-click to export	System_PLL_s...		
reset	Reset Input	Double-click to export	[clk]		
s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0000_0000	0x0000_000f
external_connection	Conduit	result_ready_pio_external_connection			
⊟ RLE_FLUSH_PIO	PIO (Parallel I/O)				
clk	Clock Input	Double-click to export	System_PLL_s...		
reset	Reset Input	Double-click to export	[clk]		
s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0000_0000	0x0000_000f
external_connection	Conduit	rle_flush_pio_external_connection			
⊟ RLE_RESET	PIO (Parallel I/O)				
clk	Clock Input	Double-click to export	System_PLL_s...		
reset	Reset Input	Double-click to export	[clk]		
s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0000_0000	0x0000_000f
external_connection	Conduit	rle_reset_external_connection			
⊟ IDATA_PIO	PIO (Parallel I/O)				
clk	Clock Input	Double-click to export	System_PLL_s...		
reset	Reset Input	Double-click to export	[clk]		
s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0000_0000	0x0000_000f
external_connection	Conduit	idata_pio_external_connection			
⊟ ODATA_PIO	PIO (Parallel I/O)				
clk	Clock Input	Double-click to export	System_PLL_s...		
reset	Reset Input	Double-click to export	[clk]		
s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0000_0000	0x0000_000f
external_connection	Conduit	odata_pio_external_connection			
⊟ FIFO_IN_FULL_PIO	PIO (Parallel I/O)				
clk	Clock Input	Double-click to export	System_PLL_s...		
reset	Reset Input	Double-click to export	[clk]		
s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0000_0000	0x0000_000f
external_connection	Conduit	fifo_in_full_pio_external_connection			
⊟ FIFO_IN_WRITE_REQ_PIO	PIO (Parallel I/O)				
clk	Clock Input	Double-click to export	System_PLL_s...		

Messages

Type	Path	Message
	8 Errors	
	Computer_System.ARM_A9_HPS.h2f_lw_axi_master	RESULT_READY_PIO.s1 (0x0..0xf) overlaps LEDs.s1 (0x0..0xf)
	Computer_System.ARM_A9_HPS.h2f_lw_axi_master	RLE_FLUSH_PIO.s1 (0x0..0xf) overlaps RESULT_READY_PIO.s1 (0x0..0xf)
	Computer_System.ARM_A9_HPS.h2f_lw_axi_master	RLE_RESET.s1 (0x0..0xf) overlaps RLE_FLUSH_PIO.s1 (0x0..0xf)
	Computer_System.ARM_A9_HPS.h2f_lw_axi_master	IDATA_PIO.s1 (0x0..0xf) overlaps RLE_RESET.s1 (0x0..0xf)
	Computer_System.ARM_A9_HPS.h2f_lw_axi_master	ODATA_PIO.s1 (0x0..0xf) overlaps IDATA_PIO.s1 (0x0..0xf)
	Computer_System.ARM_A9_HPS.h2f_lw_axi_master	FIFO_IN_FULL_PIO.s1 (0x0..0xf) overlaps ODATA_PIO.s1 (0x0..0xf)
	Computer_System.ARM_A9_HPS.h2f_lw_axi_master	FIFO_IN_WRITE_REQ_PIO.s1 (0x0..0xf) overlaps FIFO_IN_FULL_PIO.s1 (0x0..0xf)

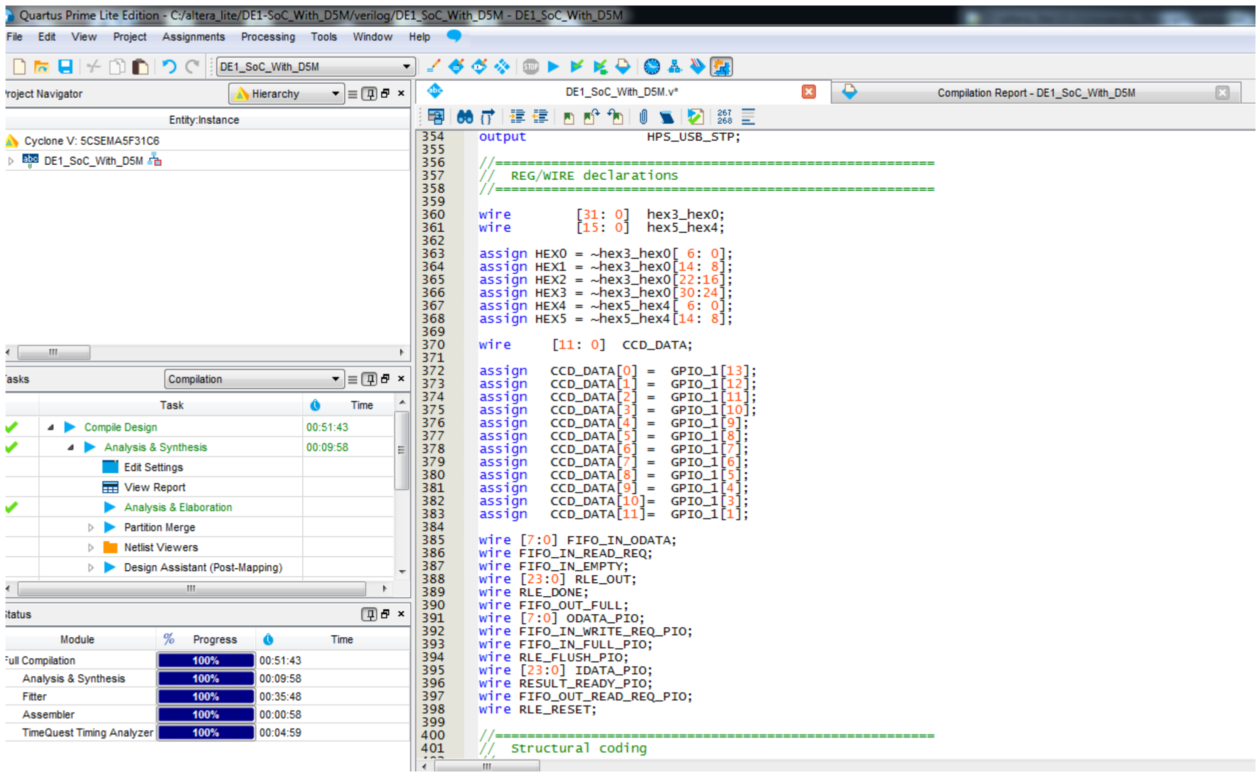
8 Errors, 2 Warnings



Name	Description	Export	Clock	Base	End	IRQ	Tags	Opcode Name
windowed_slave	Avalon Memory Mapped Slave	Double-click to export	[clock]	0xff60_0000	0xff7f_ffff			
expanded_master	Avalon Memory Mapped Master	Double-click to export	[clock]					
⊟ F2H_Mem_Window_...	Address Span Extender							
clock	Clock Input	Double-click to export	System_PL...					
reset	Reset Input	Double-click to export	[clock]					
windowed_slave	Avalon Memory Mapped Slave	Double-click to export	[clock]	0xff80_0000	0xffff_ffff			
expanded_master	Avalon Memory Mapped Master	Double-click to export	[clock]					
⊟ RESULT_READY_PIO	PIO (Parallel I/O)							
clk	Clock Input	Double-click to export	System_PL...					
reset	Reset Input	Double-click to export	[clk]					
s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0000_00d0	0x0000_00df			
external_connection	Conduit	result_ready_pio_exter...						
⊟ RLE_FLUSH_PIO	PIO (Parallel I/O)							
clk	Clock Input	Double-click to export	System_PL...					
reset	Reset Input	Double-click to export	[clk]					
s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0000_00c0	0x0000_00cf			
external_connection	Conduit	rle_flush_pio_external_...						
⊟ RLE_RESET	PIO (Parallel I/O)							
clk	Clock Input	Double-click to export	System_PL...					
reset	Reset Input	Double-click to export	[clk]					
s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0000_00b0	0x0000_00bf			
external_connection	Conduit	rle_reset_external_con...						
⊟ IDATA_PIO	PIO (Parallel I/O)							
clk	Clock Input	Double-click to export	System_PL...					
reset	Reset Input	Double-click to export	[clk]					
s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0000_00a0	0x0000_00af			
external_connection	Conduit	idata_pio_external_con...						
⊟ ODATA_PIO	PIO (Parallel I/O)							
clk	Clock Input	Double-click to export	System_PL...					
reset	Reset Input	Double-click to export	[clk]					
s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0000_0090	0x0000_009f			
external_connection	Conduit	odata_pio_external_co...						
⊟ FIFO_IN_FULL_PIO	PIO (Parallel I/O)							
clk	Clock Input	Double-click to export	System_PL...					
reset	Reset Input	Double-click to export	[clk]					
s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0000_0080	0x0000_008f			
external_connection	Conduit	fifo_in_full_pio_external...						
⊟ FIFO_IN_WRITE_REQ...	PIO (Parallel I/O)							
clk	Clock Input	Double-click to export	System_PL...					
reset	Reset Input	Double-click to export	[clk]					
s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0000_0070	0x0000_007f			
external_connection	Conduit	fifo_in_write_req_pio_e...						
⊟ FIFO_OUT_READ_RE...	PIO (Parallel I/O)							
clk	Clock Input	Double-click to export	System_PL...					
reset	Reset Input	Double-click to export	[clk]					
s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0000_0010	0x0000_001f			
external_connection	Conduit	fifo_out_read_req_pio_...						

Screenshot of QSYS before generating the design (Pay attention to the base addresses)

8. Add the wires necessary to connect your modules to each other and the ARM core in your top-level DE1_SoC_With_D5M.v file.



Quartus Prime Lite Edition - C:/altera_lite/DE1-SoC_With_D5M/verilog/DE1_SoC_With_D5M - DE1_SoC_With_D5M

Project Navigator — Hierarchy — Entity:Instance

- Cyclone V: 5CSEMA5F31C6
- DE1_SoC_With_D5M

Tasks — Compilation

Task	Time
✓ Compile Design	00:51:43
✓ Analysis & Synthesis	00:09:58
Edit Settings	
View Report	
✓ Analysis & Elaboration	
Partition Merge	
Netlist Viewers	
Design Assistant (Post-Mapping)	

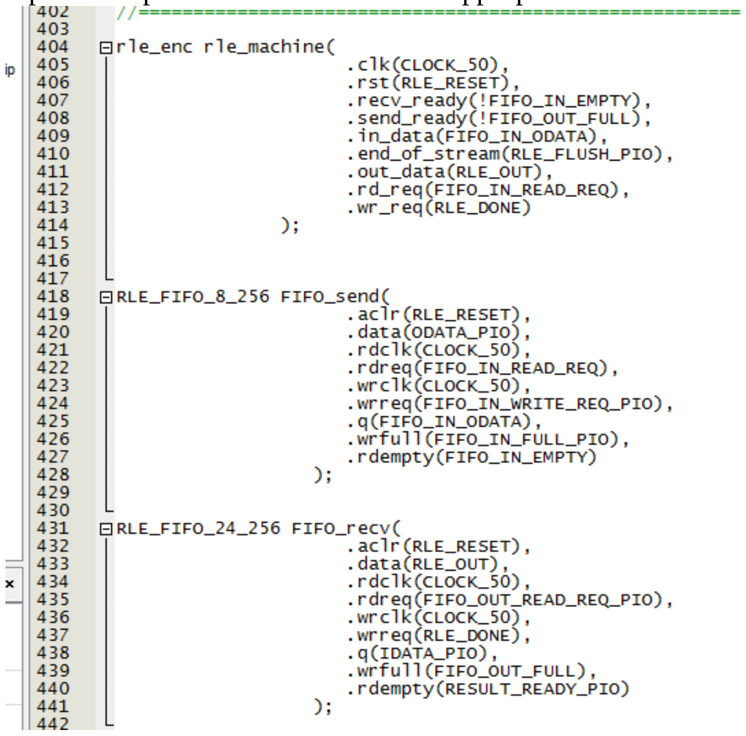
Status

Module	Progress	Time
Full Compilation	100%	00:51:43
Analysis & Synthesis	100%	00:09:58
Fitter	100%	00:35:48
Assembler	100%	00:00:58
TimeQuest Timing Analyzer	100%	00:04:59

DE1_SoC_With_D5M.v*

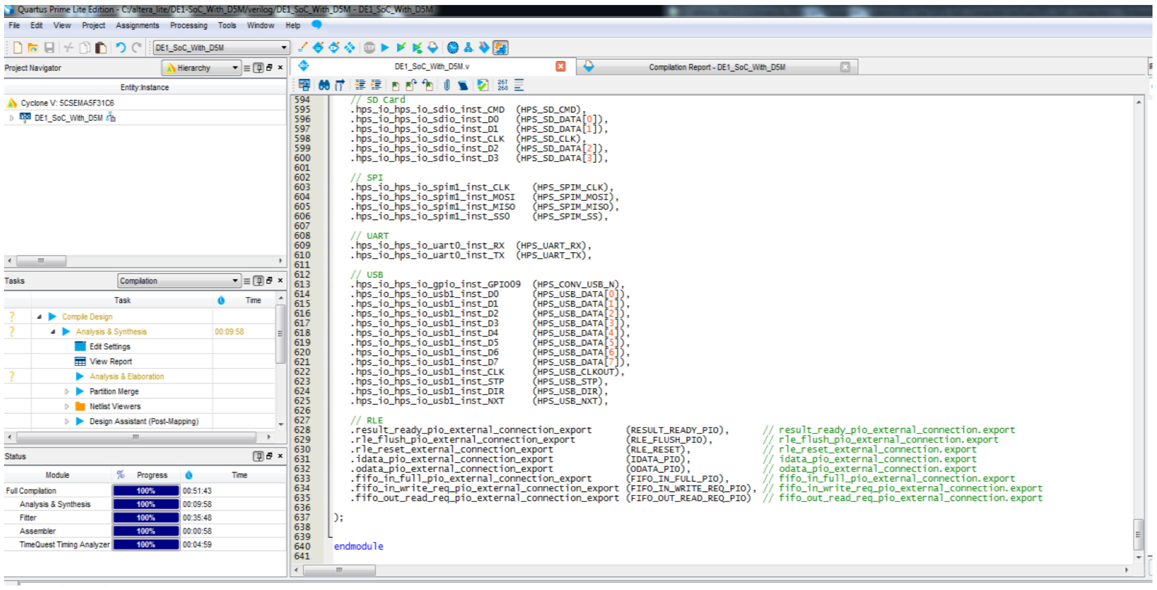
```verilog
output                  HPS_USB_STP;

//=======================================================
//  REG/WIRE declarations
//=======================================================

wire        [31: 0]  hex3_hex0;
wire        [15: 0]  hex5_hex4;

assign HEX0 = ~hex3_hex0[ 6: 0];
assign HEX1 = ~hex3_hex0[14: 8];
assign HEX2 = ~hex3_hex0[22:16];
assign HEX3 = ~hex3_hex0[30:24];
assign HEX4 = ~hex5_hex4[ 6: 0];
assign HEX5 = ~hex5_hex4[14: 8];

wire     [11: 0]  CCD_DATA;

assign   CCD_DATA[0]  =  GPIO_1[13];
assign   CCD_DATA[1]  =  GPIO_1[12];
assign   CCD_DATA[2]  =  GPIO_1[11];
assign   CCD_DATA[3]  =  GPIO_1[10];
assign   CCD_DATA[4]  =  GPIO_1[9];
assign   CCD_DATA[5]  =  GPIO_1[8];
assign   CCD_DATA[6]  =  GPIO_1[7];
assign   CCD_DATA[7]  =  GPIO_1[6];
assign   CCD_DATA[8]  =  GPIO_1[5];
assign   CCD_DATA[9]  =  GPIO_1[4];
assign   CCD_DATA[10]=  GPIO_1[3];
assign   CCD_DATA[11]=  GPIO_1[1];

wire [7:0] FIFO_IN_ODATA;
wire FIFO_IN_READ_REQ;
wire FIFO_IN_EMPTY;
wire [23:0] RLE_OUT;
wire RLE_DONE;
wire FIFO_OUT_FULL;
wire [7:0] ODATA_PIO;
wire FIFO_IN_WRITE_REQ_PIO;
wire FIFO_IN_FULL_PIO;
wire RLE_FLUSH_PIO;
wire [23:0] IDATA_PIO;
wire RESULT_READY_PIO;
wire FIFO_OUT_READ_REQ_PIO;
wire RLE_RESET;

//=======================================================
//  Structural coding
```

9. Instantiate your Verilog modules in your top-level DE1_SoC_With_D5M.v file. Connect each input or output of each module to the appropriate wire.



```verilog
//=======================================================

rle_enc rle_machine(
                    .clk(CLOCK_50),
                    .rst(RLE_RESET),
                    .recv_ready(!FIFO_IN_EMPTY),
                    .send_ready(!FIFO_OUT_FULL),
                    .in_data(FIFO_IN_ODATA),
                    .end_of_stream(RLE_FLUSH_PIO),
                    .out_data(RLE_OUT),
                    .rd_req(FIFO_IN_READ_REQ),
                    .wr_req(RLE_DONE)
            );



RLE_FIFO_8_256 FIFO_send(
                    .aclr(RLE_RESET),
                    .data(ODATA_PIO),
                    .rdclk(CLOCK_50),
                    .rdreq(FIFO_IN_READ_REQ),
                    .wrclk(CLOCK_50),
                    .wrreq(FIFO_IN_WRITE_REQ_PIO),
                    .q(FIFO_IN_ODATA),
                    .wrfull(FIFO_IN_FULL_PIO),
                    .rdempty(FIFO_IN_EMPTY)
                );

RLE_FIFO_24_256 FIFO_recv(
                    .aclr(RLE_RESET),
                    .data(RLE_OUT),
                    .rdclk(CLOCK_50),
                    .rdreq(FIFO_OUT_READ_REQ_PIO),
                    .wrclk(CLOCK_50),
                    .wrreq(RLE_DONE),
                    .q(IDATA_PIO),
                    .wrfull(FIFO_OUT_FULL),
                    .rdempty(RESULT_READY_PIO)
                );
```

10. Add the PIO connections that you added in QSYS to the system in your top-level file. The template for these connections can be found in QSYS – Generate – Show Instantiation Template.



Status

Module	% Progress	Time
Full Compilation	100%	00:51:43
Analysis & Synthesis	100%	00:09:58
Fitter	100%	00:35:48
Assembler	100%	00:00:58
TimeQuest Timing Analyzer	100%	00:04:59

```
    // SD Card
    .hps_io_hps_io_sdio_inst_CMD  (HPS_SD_CMD),
    .hps_io_hps_io_sdio_inst_D0   (HPS_SD_DATA[0]),
    .hps_io_hps_io_sdio_inst_D1   (HPS_SD_DATA[1]),
    .hps_io_hps_io_sdio_inst_CLK  (HPS_SD_CLK),
    .hps_io_hps_io_sdio_inst_D2   (HPS_SD_DATA[2]),
    .hps_io_hps_io_sdio_inst_D3   (HPS_SD_DATA[3]),

    // SPI
    .hps_io_hps_io_spim1_inst_CLK    (HPS_SPIM_CLK),
    .hps_io_hps_io_spim1_inst_MOSI   (HPS_SPIM_MOSI),
    .hps_io_hps_io_spim1_inst_MISO   (HPS_SPIM_MISO),
    .hps_io_hps_io_spim1_inst_SS0    (HPS_SPIM_SS),

    // UART
    .hps_io_hps_io_uart0_inst_RX  (HPS_UART_RX),
    .hps_io_hps_io_uart0_inst_TX  (HPS_UART_TX),

    // USB
    .hps_io_hps_io_gpio_inst_GPIO09  (HPS_CONV_USB_N),
    .hps_io_hps_io_usb1_inst_D0      (HPS_USB_DATA[0]),
    .hps_io_hps_io_usb1_inst_D1      (HPS_USB_DATA[1]),
    .hps_io_hps_io_usb1_inst_D2      (HPS_USB_DATA[2]),
    .hps_io_hps_io_usb1_inst_D3      (HPS_USB_DATA[3]),
    .hps_io_hps_io_usb1_inst_D4      (HPS_USB_DATA[4]),
    .hps_io_hps_io_usb1_inst_D5      (HPS_USB_DATA[5]),
    .hps_io_hps_io_usb1_inst_D6      (HPS_USB_DATA[6]),
    .hps_io_hps_io_usb1_inst_D7      (HPS_USB_DATA[7]),
    .hps_io_hps_io_usb1_inst_CLK     (HPS_USB_CLKOUT),
    .hps_io_hps_io_usb1_inst_STP     (HPS_USB_STP),
    .hps_io_hps_io_usb1_inst_DIR     (HPS_USB_DIR),
    .hps_io_hps_io_usb1_inst_NXT     (HPS_USB_NXT),

    // RLE
    .result_ready_pio_external_connection_export      (RESULT_READY_PIO),      // result_ready_pio_external_connection.export
    .rle_flush_pio_external_connection_export         (RLE_FLUSH_PIO),         // rle_flush_pio_external_connection.export
    .rle_reset_external_connection_export             (RLE_RESET),             // rle_reset_external_connection.export
    .idata_pio_external_connection_export             (IDATA_PIO),             // idata_pio_external_connection.export
    .odata_pio_external_connection_export             (ODATA_PIO),             // odata_pio_external_connection.export
    .fifo_in_full_pio_external_connection_export      (FIFO_IN_FULL_PIO),      // fifo_in_full_pio_external_connection.export
    .fifo_in_write_req_pio_external_connection_export (FIFO_IN_WRITE_REQ_PIO), // fifo_in_write_req_pio_external_connection.export
    .fifo_out_read_req_pio_external_connection_export (FIFO_OUT_READ_REQ_PIO)  // fifo_out_read_req_pio_external_connection.export

);


endmodule
```

11. Compile your design.

In Altera monitor program

12. Download the header files (hps_soc_system.h, socal.h, hps.h) from the website and move them to your C code folder. These header files contain all the necessary functions required for communication between ARM HPS and the FPGA. Go through these files thoroughly.

13. Open the Altera Monitor Program and download the compiled system onto the board.

14. In your C code, you will need to communicate with the RLE hardware. To do so, you will use the alt_write_byte(), alt_read_byte(), and alt_read_word() functions defined in socal.h. To write, the syntax is alt_write_byte(address, value). For the address, use ALT_FPGA_BRIDGE_LWH2F_OFST, defined in hps.h, as the offset for the base address of the lightweight bridge and add the base address of the device given in hps_soc_system.h. For example, to assert the FIFO read request signal, we would use the line

    alt_write_byte(ALT_FPGA_BRIDGE_LWH2F_OFST + FIFO_OUT_READ_REQ_PIO_BASE, 1);

15. Modify the C code to do the following.

    a. Preprocessing - Convert the captured image to one-bit-per-pixel black and white representation before compression.

    b. Communicate with the RLE hardware to perform compression. Store the resultant compressed image onto the SDRAM. Write a function in C to decompress the RLE-compressed image. Display the decompressed image along with the compression ratio.